X3D E *JAVA* NUM JOGO DE FANTASIA MEDIEVAL

X3D JAVA FUSION IN A MEDIEVAL FANTASY GAME

Nuno Miguel Pereira Martins* (martins@sal.ipg.pt) **e**
José Carlos Miranda** (jcmira@ipg.pt)

RESUMO

A apresentação de conteúdos na Internet em formato 3D é sem dúvida muito interessante e possui muitas potencialidades, especialmente se nestes mundos houver uma grande dose de interactividade. O *X3D* é um novo formato que pretende ser o *standard* para a divulgação de conteúdos 3D na Internet, substituindo, assim, o *VRML*. Para além da utilização do *X3D* convencional, existe a possibilidade de integração com a linguagem de programação externa *Java*. No âmbito deste trabalho, foi desenvolvido um protótipo de um jogo estilo *Role Playing Game* de fantasia, demonstrando, assim, as funcionalidades da integração do formato *X3D* e a linguagem de programação *Java*. A integração destas duas tecnologias foi conseguida através da interface de programação SAI (*Scene Acess Interface*) e revelou ter um futuro promissor, uma vez que permite aumentar fortemente o poder de interactividade exigido neste tipo de aplicações.

Palavras-chave. Mundos virtuais, modelação 3D, *X3D*, *Java*, *Xj3D*, Jogo RPG, Human-Animation.

ABSTRACT

The presentation of 3D contents on the Internet is doubtless very interesting and has much potential, especially if these worlds have high levels of interactivity. *X3D* is a new format that is going to be the standard for 3D contents distribution on the Internet, replacing *VRML*. Beyond the utilization of the conventional *X3D*, there is the possibility of integration with *Java*, an external programming language. In the scope of this project, was developed a prototype of a fantasy *Role Playing Game*, thus revealing the potential of the integration of *X3D* format and the *Java* programming language. The integration of these two technologies was possible through the programming interface SAI (Scene Access Interface) that showed a good outcome for it allows a considerable increase of the power of interactivity necessary for this type of applications.

Keywords. Virtual Worlds, 3D modelling, *X3D*, *Java*, *Xj3D*, *RPG*, Human-Animation.

* Licenciado Engenharia Informática na Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda, Professor da Ensiguarda – Escola Profissional da Guarda.

** Mestre em Tecnologia Multimédia, Professor da Escola de Tecnologia e Gestão do Instituto Politécnico da Guarda.

1. INTRODUÇÃO

Em 1999 foi iniciado o processo de criação do formato *X3D* (*Extensible 3D*), um padrão aberto para distribuição de conteúdos 3D na Internet. O *X3D* começou por utilizar a especificação do *VRML* (*Virtual Reality Modeling Language*) (VRML, 2007) e foi evoluindo, permitiu corrigir alguns problemas existentes com o seu antecessor. No *X3D* estão a ser aplicadas as experiências adquiridas na utilização do *VRML*, de maneira a criar um *standard* que dê resposta a um grande mercado que anseia pela popularização do 3D na *web*. A especificação do *X3D* foi aprovada pela ISO em Dezembro de 2004 (X3D, 2007).

É de evidenciar a necessidade de uma grande interactividade neste tipo de apresentações de conteúdos 3D. O *X3D* é um formato adequado para a criação e visualização de conteúdos *3D*, mas a interactividade que é possível criar sem recorrer a uma linguagem de programação externa é algo limitada. Devido à grande quantidade e diversidade de objectos no *X3D*, existem grandes potencialidades a explorar na utilização de uma linguagem de programação externa. O ambiente de programação adequado para essa manipulação é o *Java* (Harney *et al.* 2005).

Para aferir as potencialidades de integração do *X3D* com a linguagem de programação externa *Java* foi desenvolvida uma aplicação dirigida para a área concreta do entretenimento. Decidiu-se construir um jogo de *Role Playing Game* (RPG), uma vez que este tipo de aplicação exige uma grande interactividade entre o jogador e o mundo virtual. A interface de programação apropriada para estabelecer a comunicação entre as duas tecnologias é denominada *Scene Acess Interface* (SAI), a qual torna possível a troca de eventos entre a cena *X3D* e a aplicação Java, garantindo, assim, interacções muito mais poderosas.

Para colmatar a falta de informação existente nesta área tão recente, foi construída uma página *web*, onde se podem encontrar alguns exemplos de integração de *X3D* com o *Java*. Este *site* (Martins, 2007) está mais vocacionado para pessoas que já tenham alguns conhecimentos básicos de *X3D* ou VRML, na medida em que o seu objectivo principal é explicar o modo de integração do *X3D* com a linguagem Java.

Na secção 2 são apresentados alguns assuntos relacionados com o mundo virtual criado no âmbito do protótipo desenvolvido. A interface de programação que permite a comunicação entre a cena

X3D e a linguagem de programação Java é descrito na secção 3. A filosofia subjacente na programação do jogo, assim como, algumas metodologias usadas são abordadas na secção 4. Na secção 5 são apresentados os resultados obtidos e na secção 6 são expostas as conclusões e identificadas algumas linhas de orientação para trabalho futuro.

2. CRIAÇÃO DO MUNDO VIRTUAL

Nesta secção vai fazer-se referência a algumas teorias que foram tidas em conta para a criação do mundo virtual. A primeira tarefa a realizar é transpor as ideias para o papel, quer em forma de texto, quer em forma de desenhos. Estes desenhos vão funcionar como um *storyboard* e são a base para a criação do mundo virtual.

2.1 Planificação e Estratégias de Interacção

Um factor que o criador deve ter em conta na criação de um mundo virtual são as tarefas que o jogador poderá aí realizar. Um mundo virtual em que o jogador possa simplesmente navegar não será necessariamente interessante. Um princípio básico na criação de tarefas para os jogadores realizarem é que, ao completarem essas tarefas, recebam algum tipo de recompensa. Isto foi tido em conta no desenvolvimento do jogo implementado, na medida em que foram criadas algumas missões que o jogador deverá realizar ao longo da aventura.

Segundo Roehl (1994) existe uma grande necessidade de introduzir outros personagens no mundo virtual com quem o utilizador possa interagir, pois o ser humano é por natureza um ser social. Estes novos personagens devem ter algum tipo de "inteligência artificial", de maneira a tornar o mundo mais realista. No mundo construído para o jogo existem vários locais que o utilizador poderá explorar, assim como personagens com as quais pode interagir e comunicar (figura 1).

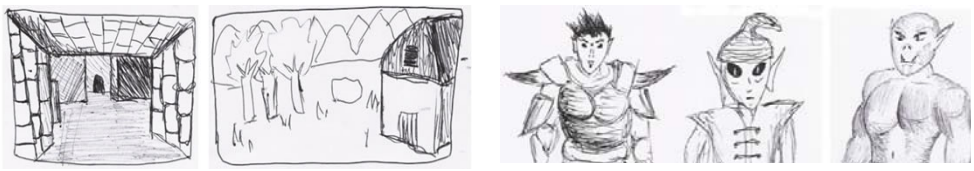

Figura 1- Desenhos de diferentes cenários e de algumas personagens do jogo

O ser humano gosta de descobrir e aprender coisas. Da mesma forma, o utilizador sentirá grande satisfação ao descobrir o modo e os "truques" necessários para atingir um objectivo, que poderá ser abrir uma porta ou aceder a um local que parecia inacessível, ou resolver *puzzles*, especialmente se depois de cada nova descoberta o utilizador for recompensado por isso. Uma maneira de manter um utilizador "preso" a um mundo virtual é recompensá-lo em diversas etapas do desenrolar da acção, de acordo com as tarefas que vai realizando.

Segundo Shneiderman (1998) uma forma de cativar os utilizadores de um mundo virtual é através da atribuição de pontuações. No jogo implementado, estas pontuações foram substituídas pelas características do personagem, cujo valor vai aumentar no decorrer do jogo à medida que o jogador vai completando as missões.

Outra característica do mundo real que igualmente se deve ter em conta num mundo virtual é a sensação de perigo e o facto de o utilizador correr riscos. Existem muitas actividades no mundo real que se tornam mais interessantes devido ao facto de existir perigo presente, como, por exemplo, andar na montanha russa, fazer *bungee jumping* ou escalar montanhas. Na figura 2 são apresentados dois exemplos de armas que foram imaginadas para o jogador utilizar nas batalhas que vão ocorrer durante o jogo. Nestas batalhas existe a possibilidade de o jogador perder pontos de vida, podendo até morrer. Esta sensação de perigo é sem dúvida muito importante num mundo virtual, especialmente quando se pretende que este seja realista e pouco monótono, juntando a isto o facto de o ser humano gostar de correr riscos e vencer adversários.

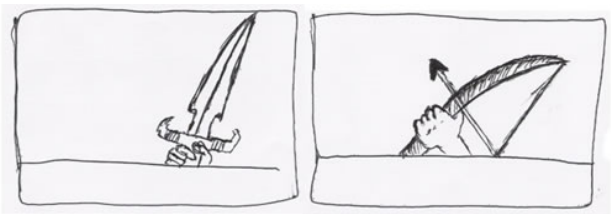

Fig. 2. Desenho de várias armas

2.2 Cenários e Objectos 3D

Após a planificação, a tarefa seguinte é a criação dos cenários e dos objectos tridimensionais que vão fazer parte do mundo virtual. Deve ser usada uma ferramenta de modelação que permita controlar a geometria ao nível dos vértices, arestas e polígonos (figura 3), de maneira a não usar simplesmente as primitivas geométricas básicas: cubo, cone, esfera, cilindro, etc..

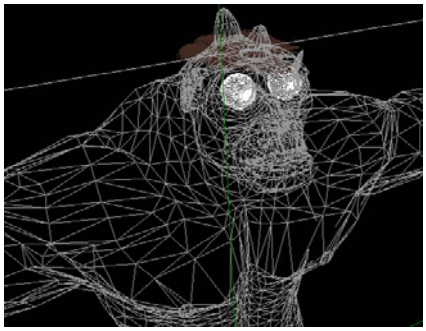

Figura 3 - Polígonos de uma personagem *X3D*

Para a modelação dos objectos que compõem o mundo virtual foram utilizadas, basicamente, duas ferramentas de modelação *3D*: o Cosmo Worlds e o Vizx3D.

O Cosmo Worlds pode ser definido como um construtor de mundos (Cosmo Worlds, 2008), tendo sido usado na criação de objectos e no respectivo posicionamento na cena, dado que possui ferramentas de precisão que permitem um grande controlo. De realçar que este modelador é um construtor de mundos *VRML*, o que implica uma posterior conversão para o formato *X3D*. Pelo contrário, o *VizX3D* (Media Machines, 2008) é um modelador específico para o formato *X3D* e, na nossa opinião, talvez seja, no momento em que se desenvolveu este trabalho, o modelador mais robusto para a criação de modelos *X3D*, uma vez que possui ferramentas de modelação bastante poderosas, tendo sido usado para a criação das geometrias mais complexas. Para além disto, foi também neste modelador que foram atribuídos os materiais e as texturas aos objectos, assim como foram criadas todas as animações. Foi igualmente no *Vizx3D* que se fez a adição de ambientes e de alguma interactividade simples.

Foram também usados objectos previamente modelados em 3D Studio MAX e mais tarde convertidos para o formato *X3D*, utilizando ferramentas de conversão, como o AccuTrans 3D (AccuTrans, 2008).

Na construção de um mundo virtual para a Internet, o processo de optimização de uma cena 3D é muito importante, porque vai permitir maiores velocidades de carregamento dos ficheiros *X3D* para o *browser*, assim como vai melhorar o seu desempenho aquando da navegação do utilizador, ou seja, aumentar a velocidade de *rendering* da cena 3D.

Na construção do mundo virtual foram usadas várias técnicas de optimização, tais como:

1. Utilizar o menor número de polígonos possível, nomeadamente eliminando aqueles que serão sempre invisíveis, seja qual for o ponto de visão. Segundo Hartman e Vogt (1998), os mundos planeados para serem vistos na Internet não devem ter mais do que, aproximadamente, 1000 triângulos visíveis a cada instante;
2. Recorrer a materiais e texturas para criar o efeito de superfícies mais complexas, sem usar muitos polígonos. Segundo Ballreich (1997), uma textura deve ser o mais pequena possível, quadrada e a sua resolução deve ser uma potência de dois;
3. Reutilizar texturas e sons sempre que possível, e com tamanhos o mais reduzidos possível;
4. Limitar o uso de luzes para manter a velocidade de *rendering* em níveis razoáveis;
5. Estruturar cuidadosamente a hierarquia dos objectos que compõem a cena, de forma a permitir que os *browsers* efectuem mais eficientemente o cálculo de visibilidade;
6. Aplicar características da computação gráfica que ajudem a aumentar a eficiência, como, por exemplo, diferentes níveis de detalhe (LOD – *level of detail*), Billboards, *Inlines* e Detecção de Colisões.

Estas técnicas são descritas, em pormenor, por Miranda e Sousa (2000).

3. INTEGRAÇÃO DE *X3D* E *JAVA*

Enquanto o *X3D* é uma linguagem adequada para a criação de objectos tridimensionais, a linguagem Java pode ser utilizada para adicionar comportamentos complexos a esses objectos. A integração destas duas tecnologias é possível através da SAI - *Scene Access*

Interface (SAI, 2004). Esta interface de programação disponibiliza um conjunto de funções que permitem a manipulação da cena *X3D* através da utilização da linguagem de programação Java.

No momento em que se realizou este trabalho, o único *browser* que permitiu a visualização de cenas *X3D* manipuladas através do Java foi o *Xj3D* (Xj3D, 2008), um *browser stand-alone* totalmente desenvolvido pelo Web3D Consortium e que tem servido como uma plataforma de testes de cenas que integram o X3D com o Java (Hudson, Couch e Matsuba, 2002; 3dtest, 2008).

Na secção seguinte vai ser explicada a forma de integrar o *X3D* com o *Java*, recorrendo, para tal, à interface de programação SAI – *Scene Access Interface.*

3.1 SAI – Scene Access Interface

A SAI é a interface da programação (API) usada para fazer a comunicação entre o *X3D* e o *Java*. Através desta ligação é possível a troca de ventos entre os nós *X3D* e a aplicação *Java*.

A interface gráfica criada para o jogo desenvolvido no âmbito deste trabalho é constituída por duas partes: o *browser X3D* e a aplicação *Java*. Enquanto o *browser* permite ao utilizador navegar num ambiente tridimensional, a interface *2D* criada em Java oferece um conjunto de ferramentas que possibilitam a interacção e manipulação dos objectos desse ambiente *3D*. Deste modo, pressionando um botão pode enviar-se um evento para a cena *X3D*, de forma a alterar as características de um determinado objecto.

A SAI permite dois tipos de acesso à cena *3D*, um tipo denominado por acesso interno e outro por acesso externo. Ambos os casos podem ser usados em conjunto, tirando desta opção as vantagens inerentes a cada um destes métodos.

No acesso interno, existe um nó de *script* dentro do ficheiro *X3D* que vai fazer a ligação com a classe Java, ou seja, é a própria cena *X3D* que, ao ser carregada num browser *X3D*, vai chamar a classe Java onde existe o código que vai efectuar as alterações pretendidas na cena (figura 4).

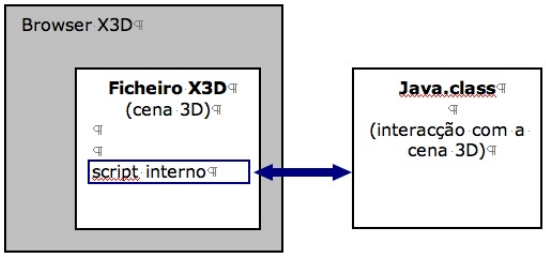

Figura 4 - Acesso interno

No acesso externo, a classe Java vai criar uma janela 3D e, de seguida, carrega a cena X3D para a janela 3D criada (figura 5**Figura**).

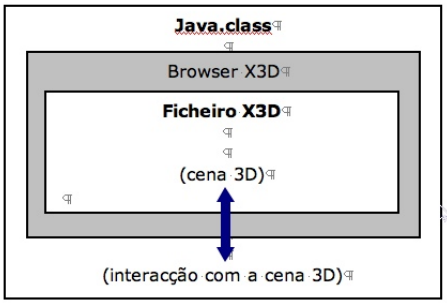

Figura 5 - Acesso externo

Neste tipo de acesso é a própria aplicação Java que vai carregar o ficheiro *X3D*. Isto torna-se bastante útil quando se pretende uma aplicação composta por duas janelas, uma referente à cena 3D e outra referente à interface *Java*, como é o caso do protótipo desenvolvido no âmbito deste trabalho.

4. Programação do Jogo

Sempre que forem necessárias a interacção e a manipulação de objectos da cena *X3D* através da linguagem de programação Java, deve-se aceder às propriedades desses objectos. Depois de se aceder a essas propriedades, pode então proceder-se à sua manipulação. De maneira a melhor se explicar o modo de funcionamento desta integração, passa-se de seguida à explicação da programação de um caso geral do jogo implementado – as batalhas do jogador com os monstros existentes no jogo.

Para simular o campo de visão do monstro, foi colocado um sensor de proximidade na área onde se encontra o monstro (figura 6).

Sempre que o jogador entrar na região demarcada por esse sensor de proximidade, é accionado um evento. A aplicação Java fica automaticamente notificada da ocorrência desse evento, desencadeando as acções necessárias, que, neste caso, seriam o início do ataque do monstro. A animação produzida é constituída por movimentos das pernas e dos braços do monstro, assim como pelos movimentos de rotação e translação na direcção do jogador.

Para controlar os movimentos de translação do monstro na direcção do jogador recorreu-se ao princípio básico de uma animação, utilizando um interpolador de posição e um relógio.

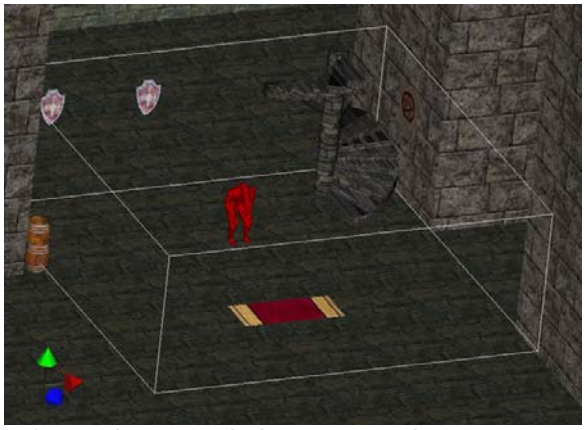

Figura 6 - Sensor de proximidade na área onde se encontra o monstro

O interpolador de posição tem unicamente dois *keyframes*: o *keyframe* inicial, constituído pelas coordenadas iniciais do monstro, e o *keyframe* final, que vai ser sempre actualizado com as coordenadas actuais do jogador, à medida que este se movimenta (figura 7).

Para controlar os movimentos de rotação do monstro, de forma a este esteja sempre "virado de frente" para o jogador, recorreu-se aos princípios básicos da geometria analítica. A representação esquemática está apresentada na figura 8.

Inicialmente, o monstro está virado numa determinada direcção, representada pelo vector $\vec{a}$. Quando o jogador entra no campo de visão do monstro, está na posição Pf. O monstro encontra-se na posição M e terá de sofrer uma rotação θ para ficar virado de frente para o jogador. Terá então de ser calculado o ângulo existente entre os vectores $\vec{a}$ e $\vec{b}$ de modo a ser conhecido o movimento que o monstro

terá de efectuar. Foi usada a fórmula apresentada na equação 1 para calcular o produto vectorial entre os vectores $\vec{a}$ e $\vec{b}$.

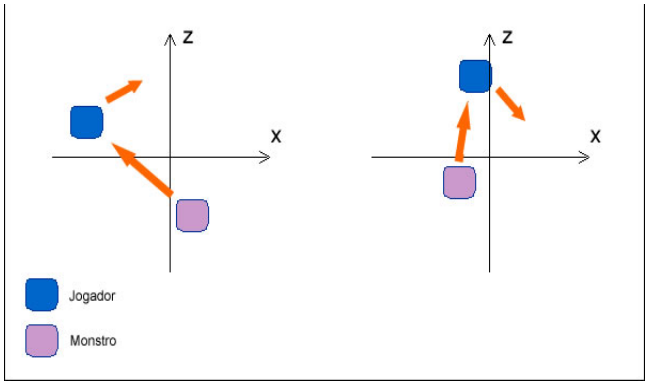

Figura 7 - Animação de translação do monstro

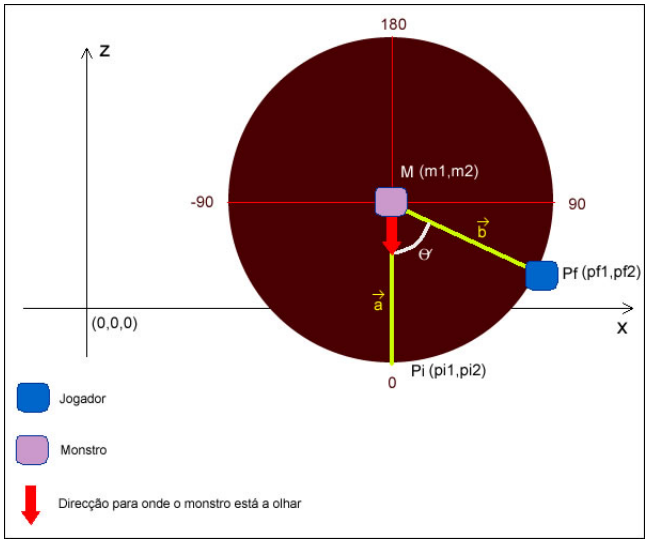

Figura 8 - Representação da rotação do monstro

$$\theta = \arccos \frac{\vec{a} \times \vec{b}}{\left\|\vec{a}\right\| \times \left\|\vec{b}\right\|} \qquad (1)$$

Em que $\|\vec{a}\|$ é a norma do vector $\vec{a}$, $\|\vec{b}\|$ é a norma do vector $\vec{b}$ e θ é o ângulo entre os dois vectores

O cálculo do ângulo θ é realizado continuamente e aplicado ao sistema de coordenadas do monstro, garantindo, assim, que o monstro fica sempre virado de frente para o jogador.

Posteriormente, torna-se necessário controlar quando o jogador está próximo do monstro, sendo esta a situação em que o monstro iniciará o seu ataque, podendo até causar dano. Para verificar esta situação, colocou-se um sensor de proximidade junto do corpo do monstro. Este sensor de proximidade está a envolver todo o monstro e encontra-se dentro do grupo que faz parte da animação do mesmo, para que, quando este se desloca, transporte o sensor de proximidade consigo (figura 9**Figura**).

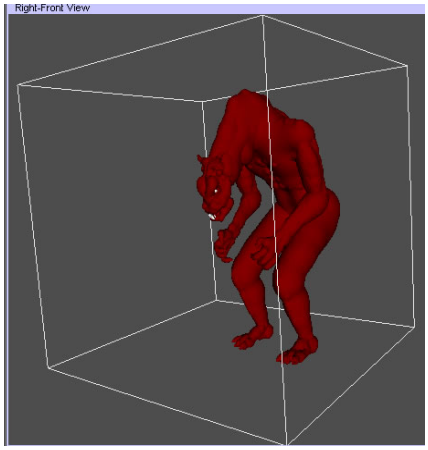

Figura 9 - Sensor de proximidade do monstro

A partir do momento em que o jogador entra na região definida por este segundo sensor de proximidade, será, então, accionado outro evento que indica ao programa que os ataques podem começar. Assim, são iniciadas as animações de ataque do monstro e do jogador. O dano causado por estes ataques é controlado por fórmulas específicas, que têm em conta as características físicas do jogador, o tipo de arma equipada e as particularidades do monstro.

Uma batalha termina quando os pontos de vida do jogador ou do monstro chegarem a zero, considerando-se que este morreu. Deste modo vão ser controlados, as batalhas e o dano causado, quer pelo monstro, quer pelo jogador.

Depois de visto este exemplo, pode entender-se um pouco melhor a filosofia na programação deste jogo. O mundo criado é

constituído por um grande número de sensores, como, por exemplo, sensores de proximidade, sensores de toque, sensores de visibilidade, entre outros. Desta maneira, pretende-se que cada sensor vá notificar a aplicação Java das interacções do jogador, que por sua vez vai desencadear uma determinada acção sobre o objecto específico do mundo 3D.

5. Resultados

Para aferir as potencialidades de integração da tecnologia X3D com a linguagem de programação Java, foi desenvolvido um protótipo de um jogo RPG de fantasia medieval. A interface gráfica é constituído por duas partes: o *browser* X3D e a aplicação *Java* (figura 10). Enquanto o *browser* permite ao utilizador navegar num ambiente tridimensional, a interface criada em Java oferece um conjunto de ferramentas que possibilitam a interacção e a manipulação dos objectos desse ambiente *3D*.

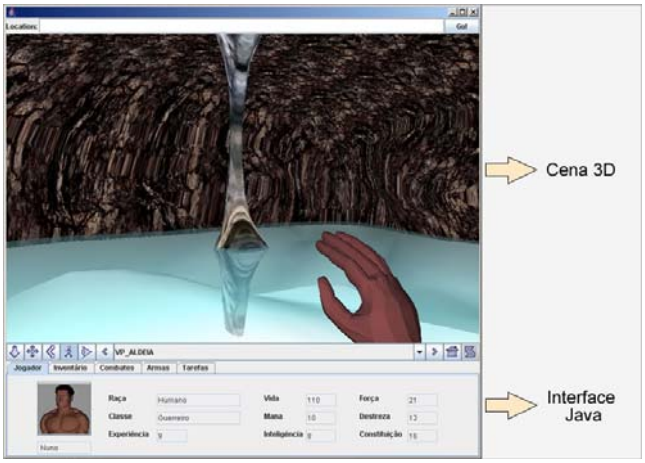

Figura 10 – Interface gráfico constituído por dois componentes: cena 3D e aplicação Java

Na parte inicial do jogo, o utilizador deverá seleccionar e configurar as características da sua personagem, como, por exemplo, a raça, a cor, a força, a destreza, a inteligência, etc.. Nesta etapa inicial existem grandes possibilidades de interacção do utilizador com a cena tridimensional. Através de uma interface desenvolvida em Java, é possível não só manipular algumas características da personagem, como também adicionar novos elementos na cena *3D*. Posteriormente,

o jogador poderá explorar um mundo virtual num ambiente de fantasia medieval. O mundo criado é constituído por diversos ambientes, tais como montanhas, grutas, aldeias, castelos e florestas. O jogador deverá realizar algumas tarefas e explorar o mundo da maneira que pretender. Não existe uma linha de jogo fixa, o que significa que o jogador possui um elevado grau de liberdade na sua exploração. Ele poderá decidir com que personagens falar, quais os monstros a atacar, quando deverá fugir, etc.. Nesta fase do jogo existe igualmente uma interface *2D* desenvolvida em *Java* que permite a interacção com a cena *X3D*.

Na figura 11 são apresentadas algumas imagens do protótipo desenvolvido no âmbito deste projecto.

Para executar o som na cena 3D foi necessária a instalação do OpenAL (2008), uma biblioteca muito utilizada em aplicações e jogos que necessitem de áudio *3D*.

Neste tipo de jogos não se pretende que o jogador atinja um determinado ponto para a passar ao nível seguinte, mas sim que o jogador explore o mundo e realize tarefas, de maneira a aumentar as suas capacidades e habilidades, e que, acima de tudo, goste da experiência de explorar o mundo virtual.

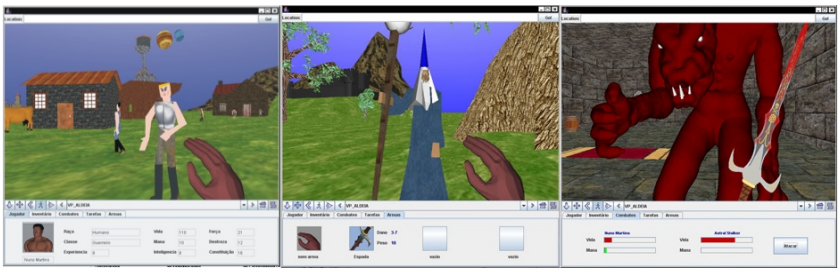

Figura 11 - Imagens do jogo desenvolvido

6. CONCLUSÃO

Para a realização deste trabalho optou-se pela utilização de uma tecnologia *3D open source* - o *X3D*. A escolha do *X3D* em detrimento do *VRML* parece lógica, tendo em conta que aquele é o sucessor do último, possuindo, entre outras vantagens, uma codificação da cena muito mais robusta. Os objectos da cena *3D* são, agora, codificados em *XML*, ao invés de texto, como acontecia com o *VRML*.

O objectivo imediato deste projecto não seria a construção de um jogo, mas sim a criação de uma aplicação que demonstrasse as capacidades e potencialidades da ligação entre um mundo *X3D* e uma aplicação Java. No entanto, pensa-se que com o jogo implementado ficam demonstradas bastantes características desta integração. A capacidade de recorrer a uma linguagem de programação externa para adicionar comportamentos específicos aos objectos de uma cena *3D* veio aumentar o poder de interactividade existente, tornando, assim, o *X3D* muito mais poderoso.

Embora o protótipo desenvolvido fosse dirigido para a área concreta do entretenimento, a integração destas duas tecnologias pode ser utilizada em diversas áreas de aplicação, como, por exemplo, arquitectura, medicina, ensino e publicidade, entre outras.

O *browser* utilizado, *Xj3D*, funciona bem na fusão do *X3D* com o Java e suporta a maior parte das características do *X3D*. No entanto, apresenta ainda alguma instabilidade, devido ao facto de estar ainda em desenvolvimento. Espera-se que seja brevemente lançada uma nova versão deste *browser*, com muitos dos seus problemas resolvidos. É, ainda, de realçar a necessidade de adicionar tempo extra para uma correcta preparação do computador, caso contrário não é possível a comunicação entre o *X3D* e o *Java*.

A principal dificuldade encontrada no desenvolvimento deste trabalho foi a falta de informação existente sobre o tema. As informações disponíveis não eram muito claras, especialmente na forma de integração do *X3D* com o Java. De modo a colmatar a falta de informação nesta área tão recente e, de alguma forma, contribuir para a evolução e divulgação desta tecnologia 3D *standard*, foi criada uma página *Web* composta por exemplos práticos. Esta página está *online* e pode ser consultada a partir do *site* oficial do *Xj3D*, na sua secção de tutoriais (2008).

Trabalho futuro

A implementação de um jogo não é uma tarefa simples, na medida em que pressupõe a intervenção de uma equipa de trabalho, cada elemento com uma função específica. Muito do trabalho futuro será a criação de novas áreas para o jogo, assim como a implementação de novas missões e objectivos para o jogador cumprir. Adicionar inteligência aos personagens de modo a tornar o mundo mais realista é também outro objectivo futuro.

Durante a realização deste projecto a impossibilidade de utilizar a SAI numa aplicação *online*, devido ao facto de o *Xj3D* ser um *browser*

stand-alone. Sendo assim, outro objectivo futuro passará pela disponibilização na *Internet* do jogo criado.

BIBLIOGRAFIA

Ballreich, C. (1997); *Late Night VRML 2.0 with Java*; Ziff-David Press; MacMillan Computer Publishing.
Couch, Justin et al. (1997); *Late night VRML 2.0 with Java*; Emeryville; Ziff-Davis Press.
Harney, J. *et al.* (2005); "Visualizing Information Using SVG and X3D: X3D Graphics, Java and Semantic Web" in *LNCS V*; Geroimenko and C. Chen eds.; vol. 5555; Springer, Heldelberg; 10-10.
Hartman, Jed; Vogt, Wendy (1998) *Cosmo Worlds 2.0 User's Guide*; Silicon Graphics, Inc.; EUA.
Hudson, A.; Couch, J.; Matsuba, S. (2002); "The XJ3D Browser: Community-Based 3D Software Development"; *ACM SIGGRAPH 2002 Conference*; ACM Press; Texas; 327-327.
Miranda, José; Sousa, A. (2000); "Urbanismo e Espaços Virtuais – Divulgação e discussão na Comunidade"; *Virtual- Revista Electrónica de Visualização, Sistemas Interactivos e Reconhecimento de Padrões*; Faculdade de Engenharia da Universidade do Porto.
Roehl, Bernie (1994); *Playing God – Creating Virtual Worlds with Rend386*; EUA; WAIT Group Press, s.d.
Shneiderman, Ben; Plaisant, C. (1998); *Designing the user interface – Strategies for Effective Human-Computer Interaction*; Addison Wesley, EUA.

Sítios:
AccuTrans (2008); AccuTrans 3D by MicroMouse Productions;
http://www.micromouse.ca/
Cosmo Worlds (2008); Cosmo Worlds 2.0 gives 3-D Web authoring a boost,
http://www.infoworld.com/cgi-bin/displayArchive.pl?/98/20/i12-20.81.htm
Scene Access Interface (SAI) (2004); Extensible 3D (X3D) - Part 2: ISO/IEC 19775-2
http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/
Martins, Nuno (2007); *X3D* – Integração com *Java*; Instituto Politécnico da Guarda;
http://www.sal.ipg.pt/user/estg/martins/x3d.htm.
Media Machines (2008); Media Machines - Developer Resources and Forums;
http://www.mediamachines.com/developer.php
openAL (2008) Cross-Platform 3D Audio; http://www.openal.org/
Web3D (2005); Web3D News & Events; (20/11/2005) San Francisco;
http://www.web3d.org/news/archives/
VRML – Virtual Reality Modelling Language (2007);
http://www.web3d.org/x3d/specifications/vrml/VRML1.0/index.html
X3D (2007); X3D and Related Specifications; http://www.web3d.org/xed/specifications/.
Xj3D (2008); Java Based X3D Toolkit5 and X3D Browser; http://www.Xj3d.org.
Xj3D Tutorials (2008); Portuguese SAI Tutorial; http://www.xj3d.org/tutorials/index.html
3dtest (2008) ; 3D Temps reel et interactive; http://www.3d-test.com/interviews/xj3d_1.htm